BACKSLASH DESIGN

第11回　設計図フォーラム

コード起点プロダクトライン開発

2025年7月11日

山田 悠貴

ビースラッシュ株式会社

本日の内容

BACKSLASH DESIGN

1．ソフトウェアの資産化

2．７STEPコード起点プロダクトライン開発

3．AtScopeを使った手順紹介

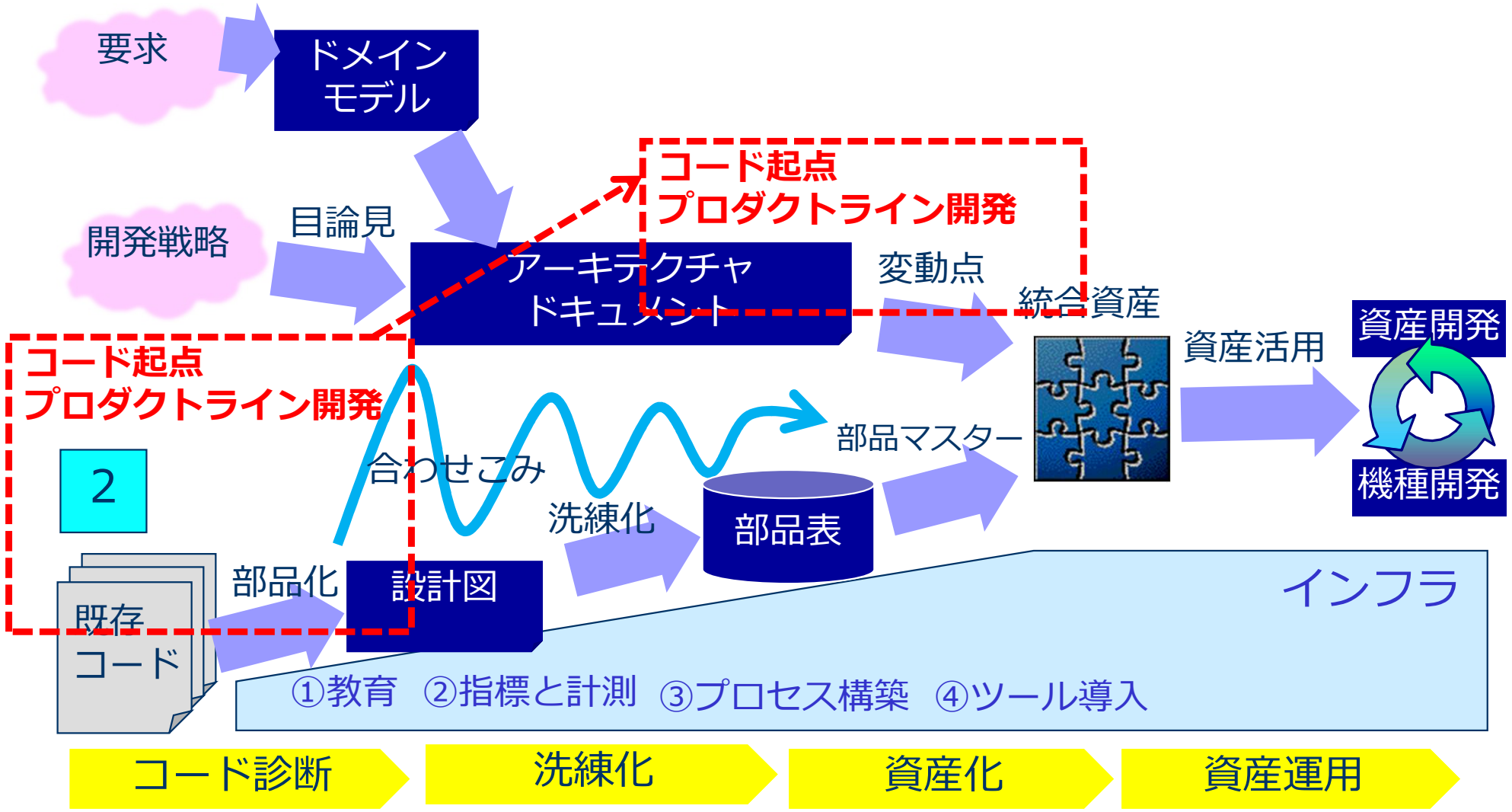
4．まとめ：コード起点での資産化

BACKSLASH DESIGN

1. ソフトウェアの資産化

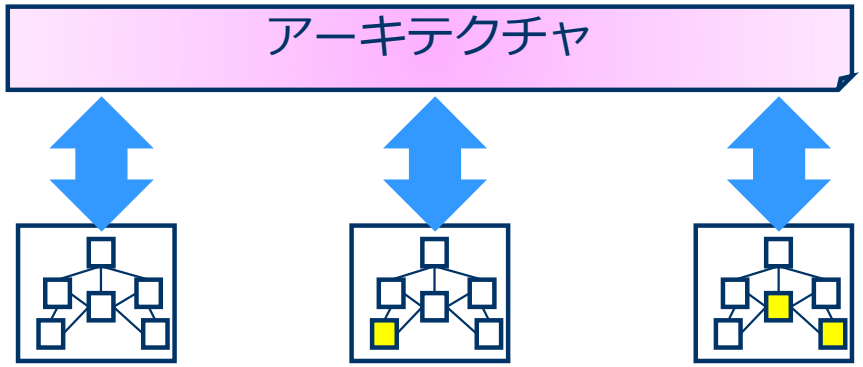
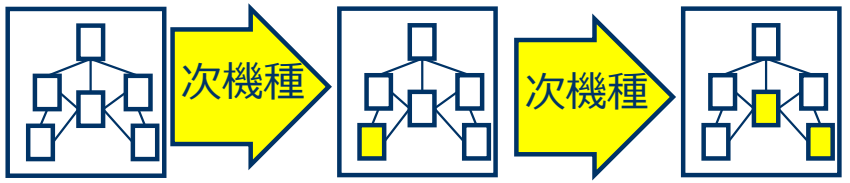
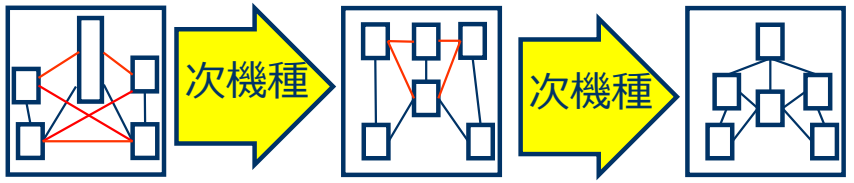
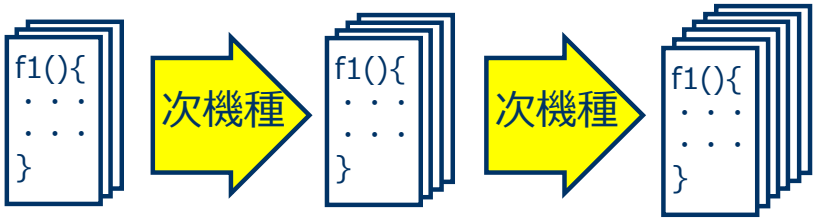
ソフトウェアアーキテクト活動

アーキテクチャとコードの同期で戦略的開発を実現



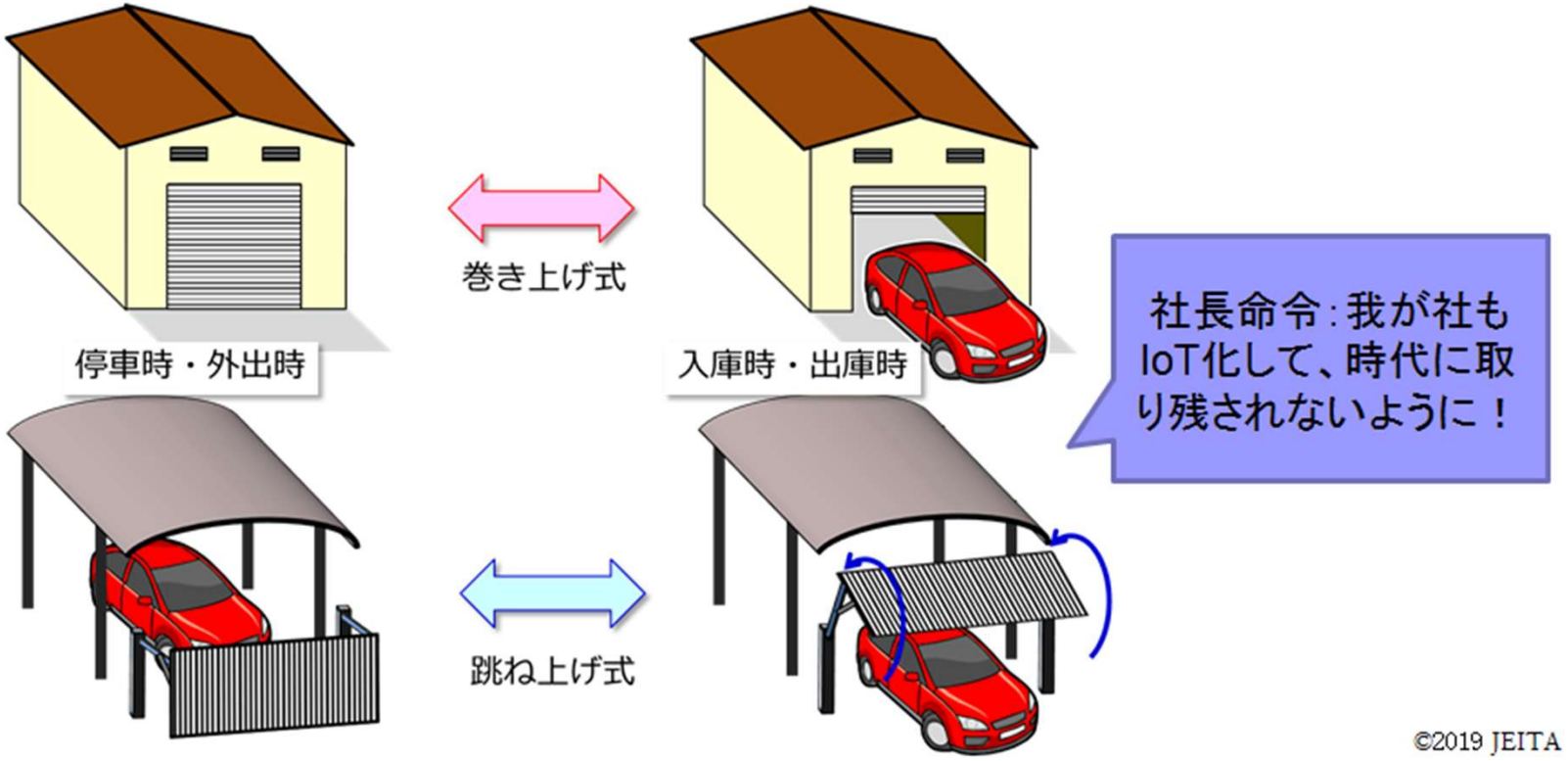
ソフトウェアの資産化

アーキテクト活動での
戦略的なソフトウェアの資産化

Lv	再利用方式	概念図	重要な施策
4	プロダクトライン		アーキテクチャ スコーピング 変動点管理
3	プラットフォーム （部品化再利用）		部品化 設計図 設計技法
2	リファクタリング	 赤線：双方向依存	高凝集 疎結合
1	ソースコード流用		人海戦術（時間をかけての引き継ぎ） 水際作戦（大量のテストで品質確保）

例題：JEITAスマートガレージ

- 現行システムの目的：クルマから降りることなく、ガレージを開閉する
  - 雨に濡れないで、ガレージを開閉する
  - クルマを止めてアイドリングすることなく、ガレージを開閉する
  - ガレージドアを開けたまま／施錠しないまま外出するのは防犯上望ましくない



https://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=1155&ca=1#top

BACKSLASH DESIGN

2．7STEPコード起点プロダクトライン開発

7STEPコード起点プロダクトライン開発

BACKSLASH DESIGN

STEP	アクティビティ
1	全体俯瞰
2	単方向依存
3	提供インタフェース定義
4	スコーピング
5	フィーチャ定義
6	変動点マップ
7	可変性実装

BACKSLASH DESIGN

2-1. 全体俯瞰

動くコードの構造を見える化する

ソースコードを局所的に見えても構造は分からない
ファイル単位に呼出関係を書いても粒度が細かすぎる
↓
フォルダ単位の構造図を作る！

- アーキテクチャに従ってフォルダを作り、全体を俯瞰する

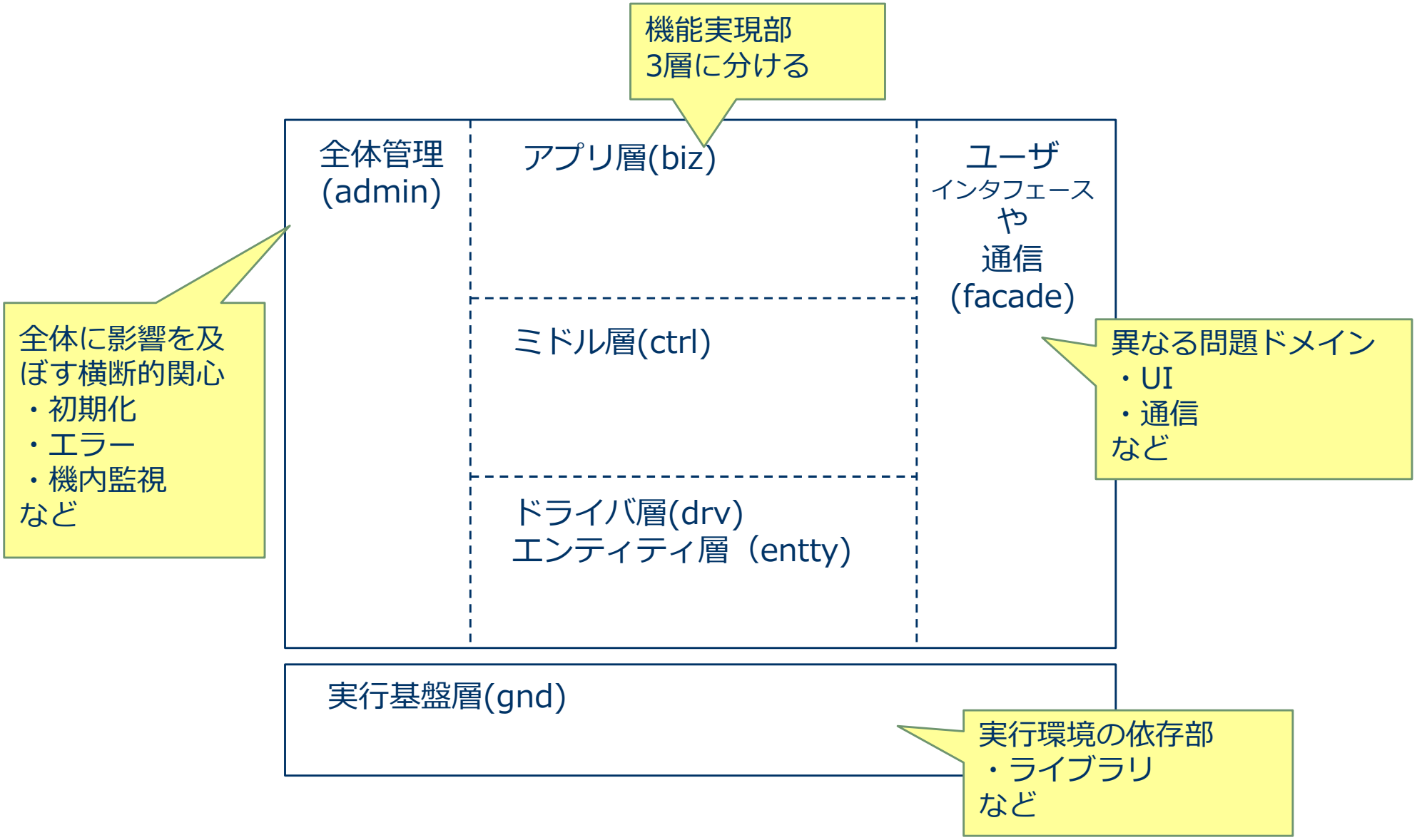
具体的な手段は

- フォルダ形状でアーキテクチャの意図を表現
  - 水平垂直分割
  - 「A2Gアーキテクチャ」と呼んでいます（次シート）
- フォルダ単位の図面化
  - AtScopeはコンパイルできなくても図面化します！
  - フォルダを作って、コードを移動することで、構造図がでます

静的構造の型：A2Gアーキテクチャ

BACKSLASH DESIGN

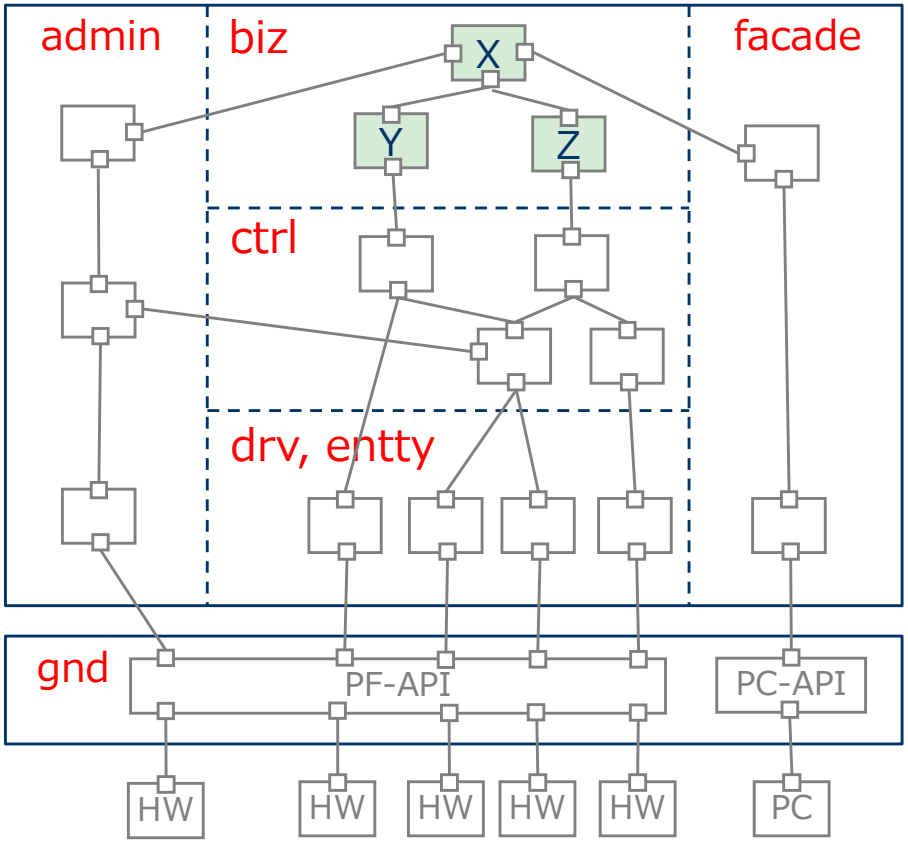
- 水平垂直分割する

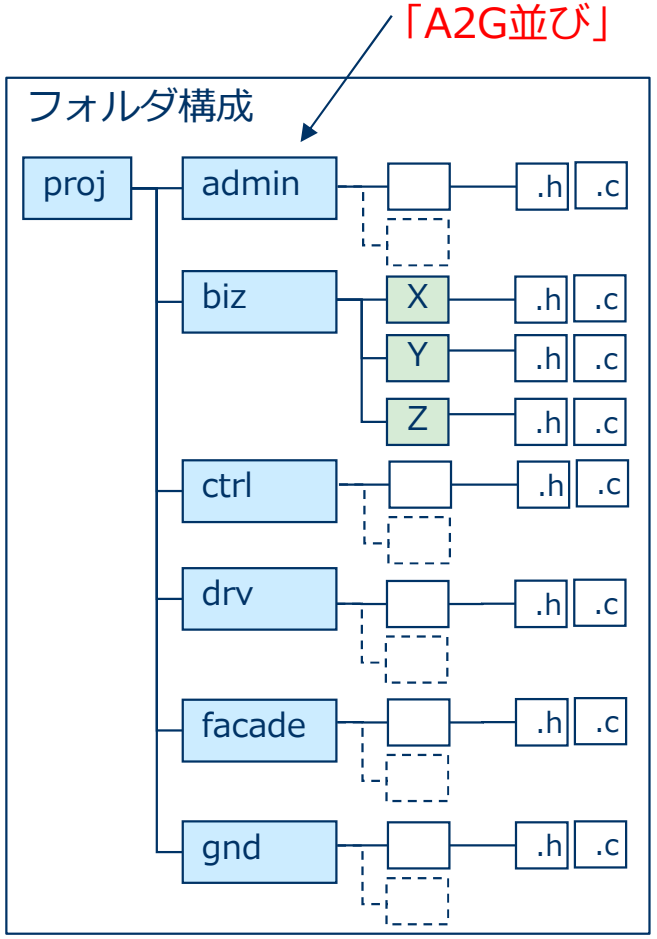


A2Gアーキテクチャとフォルダ構成

- システム全体を水平垂直分割して、コンポーネントを置く
  - 中央部をアプリ層(biz)－ミドル層(ctrl)ードライバ層(drv ,entty)
  - 左側に横断的関心(admin)、右側に異なるドメイン(façade)
  - 下部にライブラリ利用(gnd)

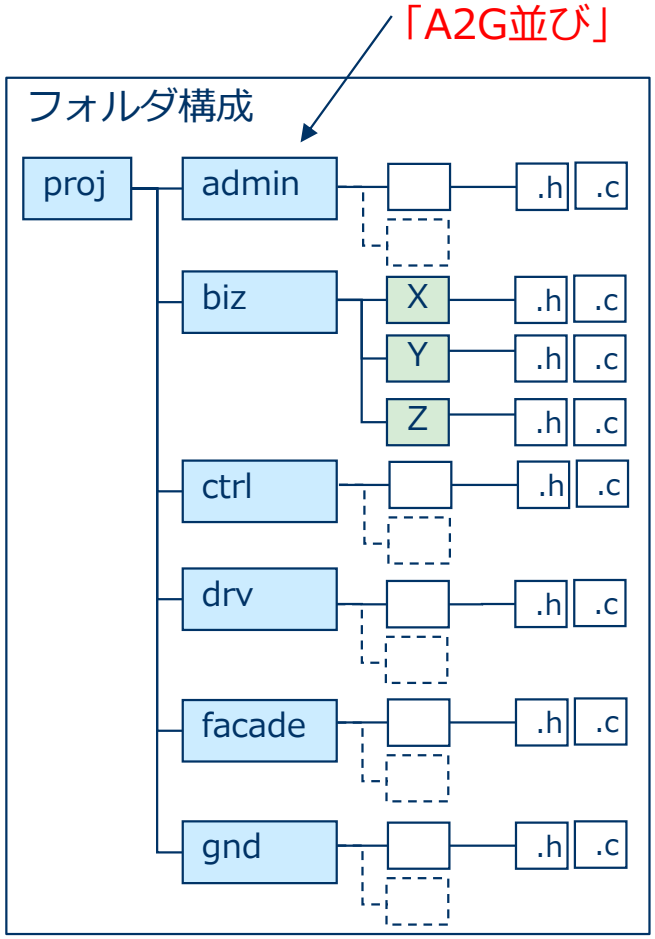
drvはハードウェア(HW)呼出し
enttyはデータベース(DB)呼出し



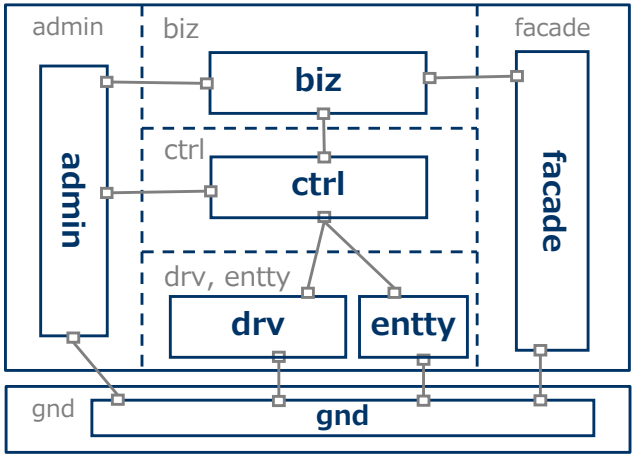


フォルダ単位の静的構造図

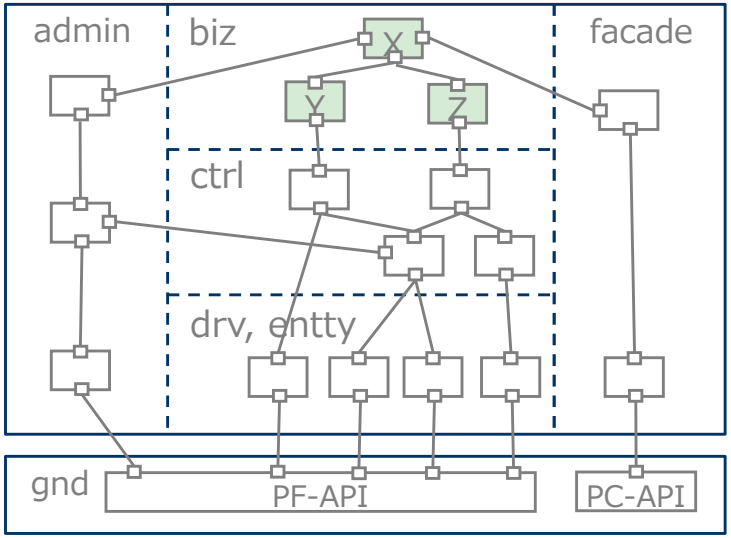
- フォルダ指定場所で粒度を変えることができる
  - ルート直下フォルダでは、全体を7個のコンポーネントで図面化
  - 2層目フォルダでは、より粒度の小さい単位で図面化



ルート直下フォルダ



2層目フォルダ



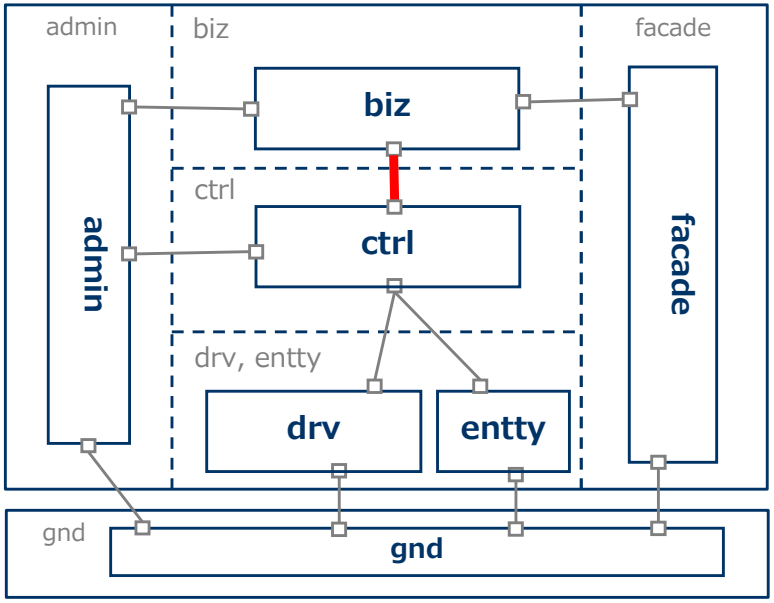
BACKSLASH DESIGN

2-2. 単方向依存

アーキテクチャにインパクトを与えるコードを改善

- 場当たり的にコードをリファクタリングしても品質はそれほど上がりません

- アーキテクチャにインパクトを与える個所を改善
  - フォルダ単位の双方向依存を、単方向依存へ
  - 単方向依存にすることで、置換可能になります

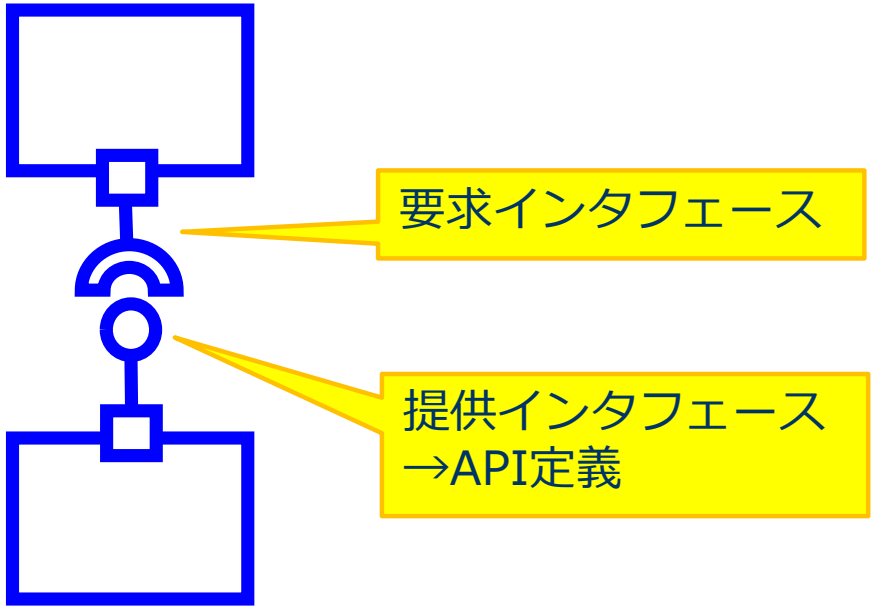


BACKSLASH DESIGN

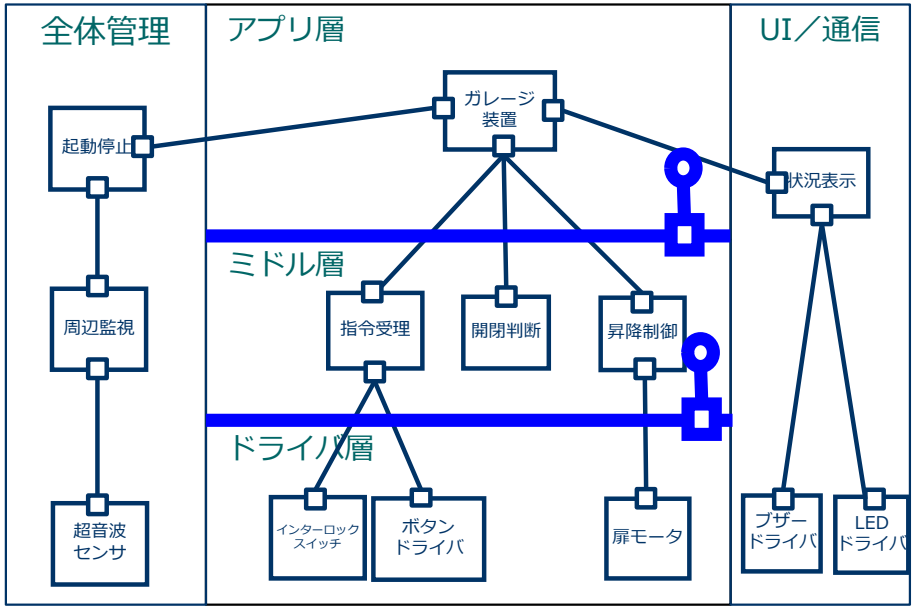
2-3. インタフェース定義

フォルダ間のインタフェースを定義

- フォルダ間の関数呼出しの集合をインタフェースとして定義します
  - 提供インタフェース群をAPIとして定義
- 列挙してみることで均質さを向上します
  - 粒度の異なる呼び出しが見つかる



レイヤー毎の
提供インタフェース



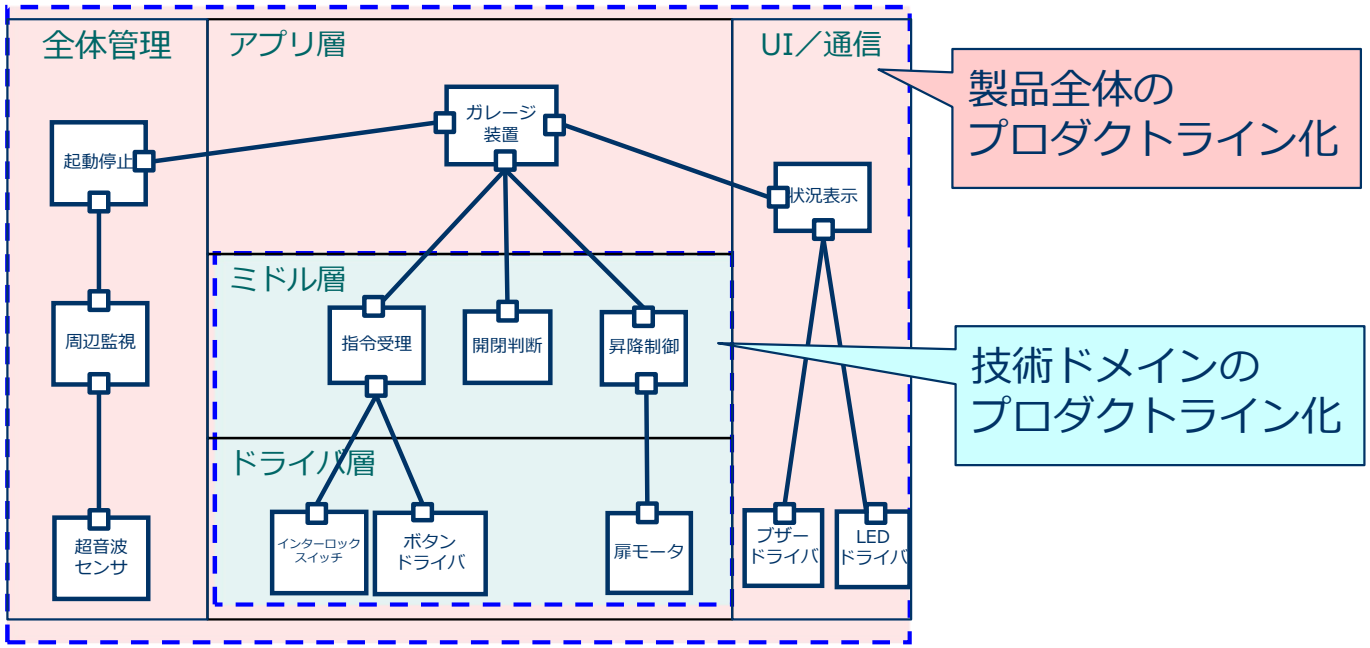
BACKSLASH DESIGN

2-4. スコーピング

プロダクトライン化する単位を決める

コンポーネント構造図を括って単位（スコープ）を決める

- 製品全体のプロダクトライン化
- 技術ドメインのプロダクトライン化

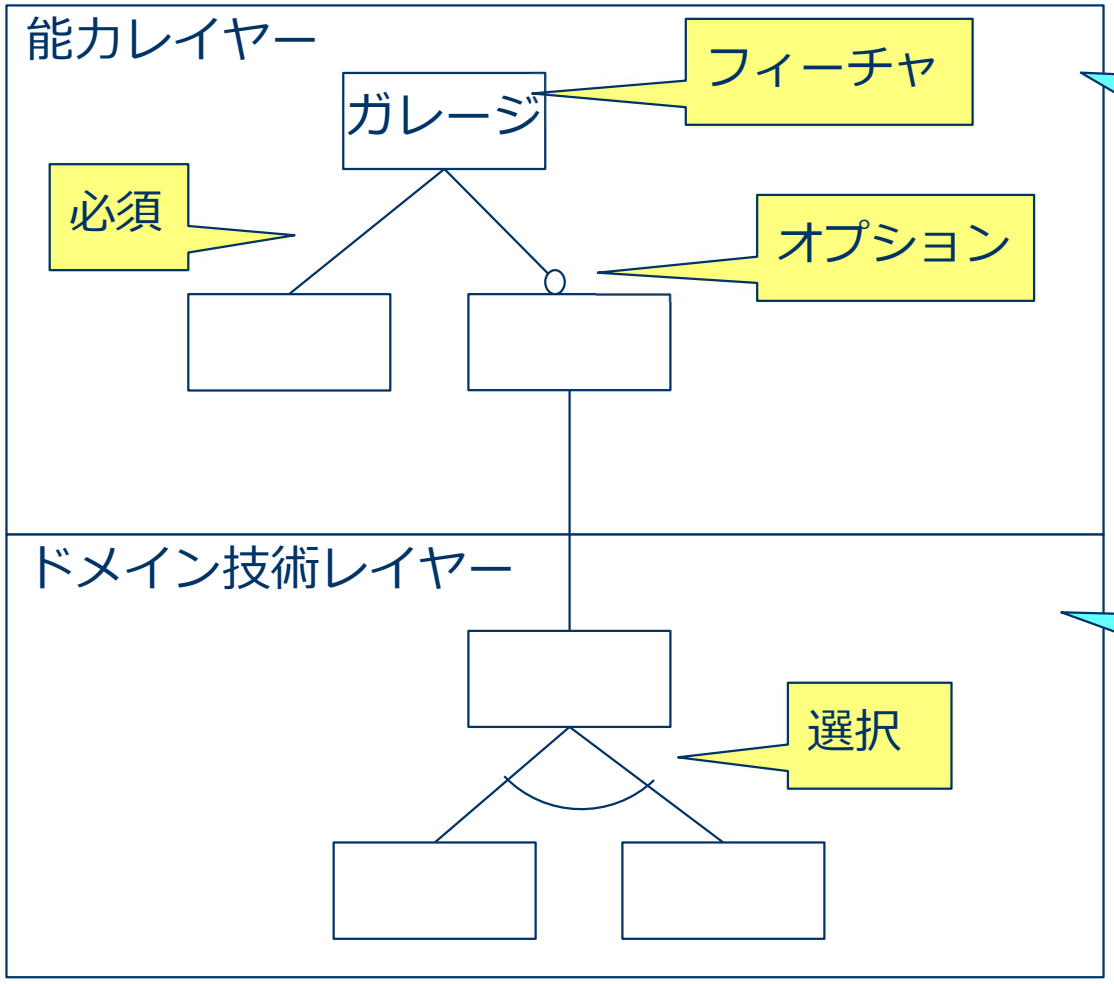


BACKSLASH DESIGN

2-5. フィーチャ定義

変動点管理：フィーチャモデル

- オプションや選択を図解する



カタログスペック
レベルの変動点

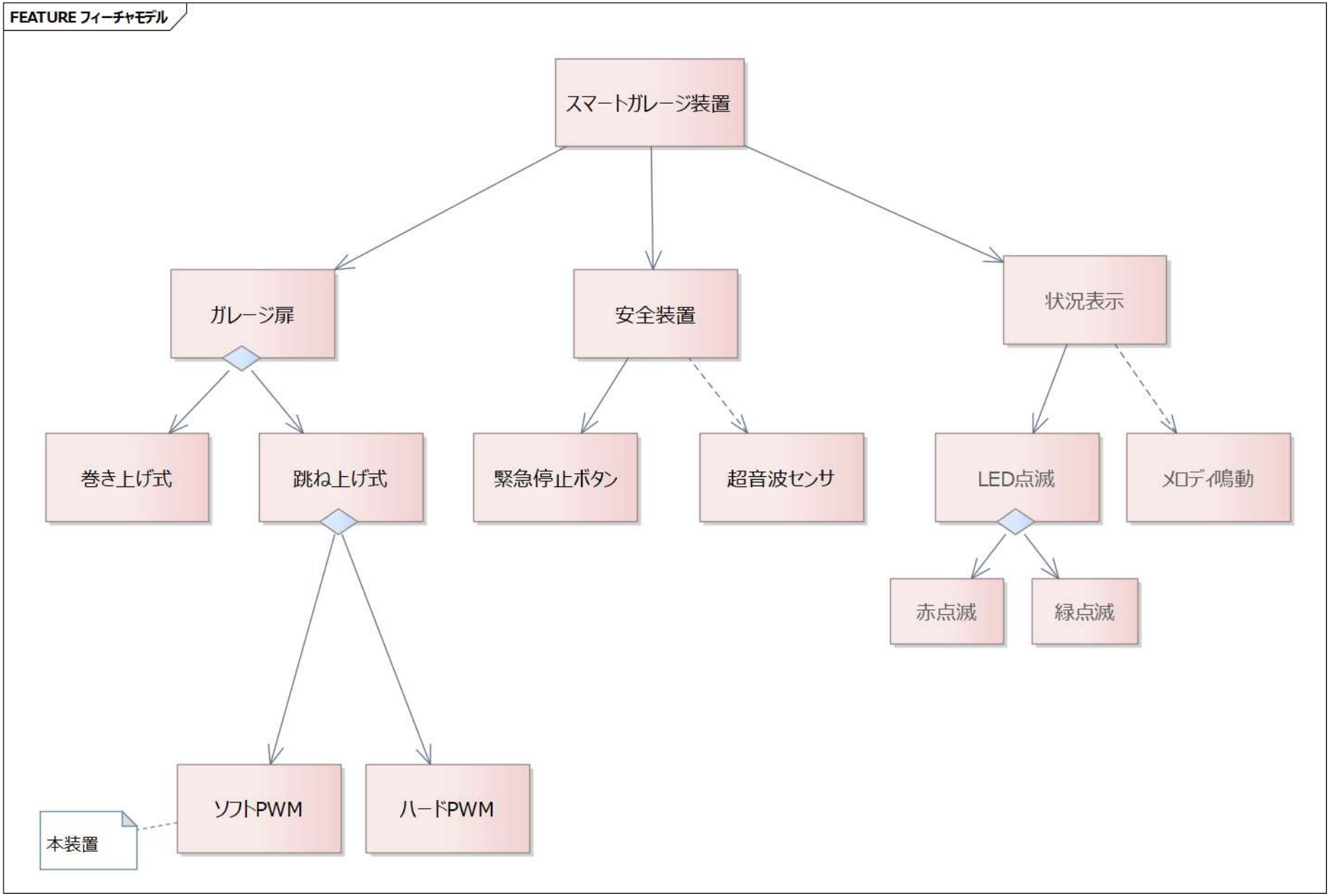
エンドユーザから見えるサービス。
利用者及び管理者が意識すること。

動作レベルの変動点

ドメイン特有のアルゴリズム、
及び、規格や法令、

こちらは数が多くなるので、
図（フィーチャモデル）ではなく、
表（マトリクス）で管理することが多い。

フィーチャモデルの例


FEATURE フィーチャモデル
スマートガレージ装置
ガレージ扉
安全装置
状況表示
巻き上げ式
跳ね上げ式
緊急停止ボタン
超音波センサ
LED点滅
メロディ鳴動
赤点滅
緑点滅
ソフトPWM
ハードPWM
本装置

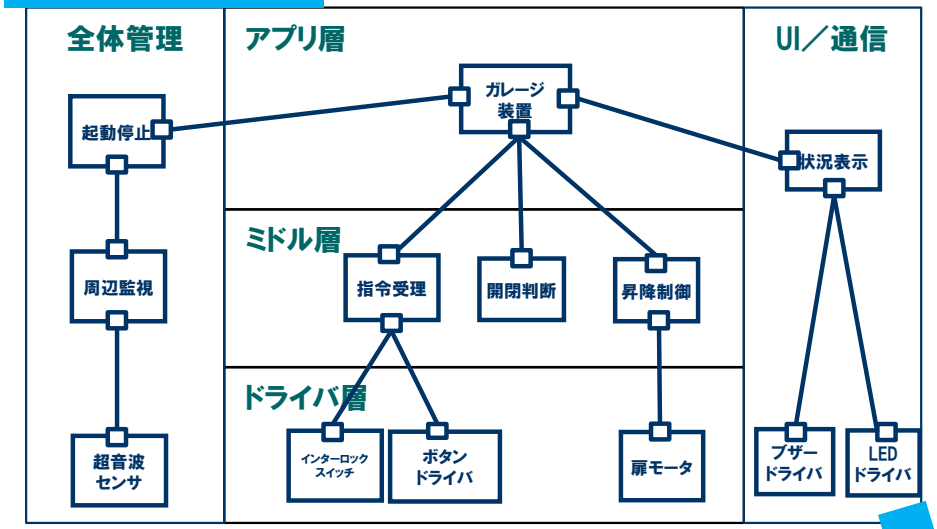
BACKSLASH DESIGN

2-6. 変動点マップ

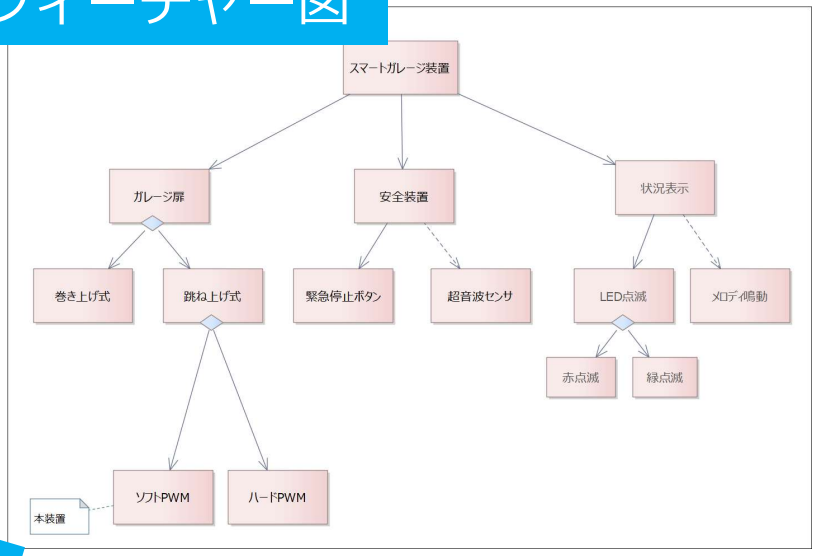
変動点マップ

- どのコンポーネントにどの変動点が含まれているか

静的構造図



フィーチャー図



変動点マップ

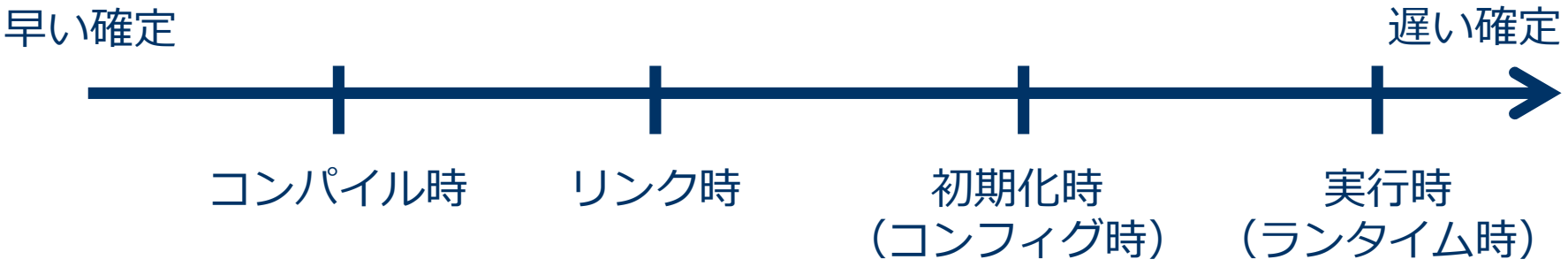
コンポーネント / フィーチャ	アプリ層	ミドル層			ドライバ層			全体管理			UI		
	ガレージ装置	指令受理	開閉判断	昇降制御	インターロックスイッチ	ボタンドライバ	扉モータ	起動停止	周辺監視	超音波センサ	状況表示	ブザードライバ	LEDドライバ
ガレージ扉				○									
跳ね上げ式							○						
安全装置										○			
状況表示											○	○	
LED点滅													○

BACKSLASH DESIGN

2-7. 可変性実装

バインディングタイム（機能確定のタイミング）

- いつ機能を確定するか
  - 遅い確定の方が、機能の柔軟さを作り込める
    - Late Binding
    - 関数ポインタの差し替えなどで実現
  - 早い確定の方が、テストがしやすく、品質の早期安定へ



変動点の設計

確定時	方法	概要
コンパイル時	別名ファイル	Makefileでコンパイル対象のファイルを差し替える
	条件コンパイル	コンパイルスイッチでコンパイル対象の領域を切り替える
	フックメソッド	変動点の関数を集めたファイルを差し替える
リンク時	実行ファイル	分割コンパイルして、実行ファイルをリンク時に変える
初期化時	コンフィグ設定	装置構成などのコンフィグ設定で、動作するプログラムを切り替える
	ファクトリ	機能設定などに応じて、クラスから生成するインスタンスを変える（オブジェクト指向的）
実行時	仮想関数	関数ポインタ（コールバック関数）のアドレスを、プログラムで切り替える。

BACKSLASH DESIGN

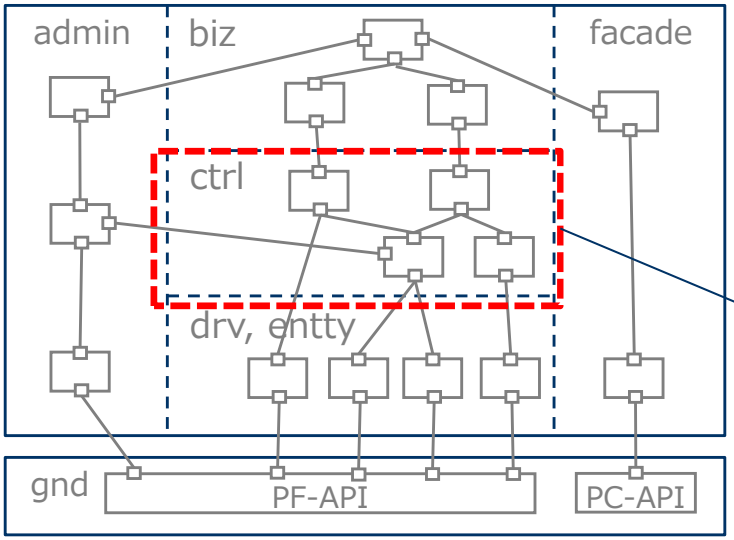
3．AtScopeを使った手順紹介

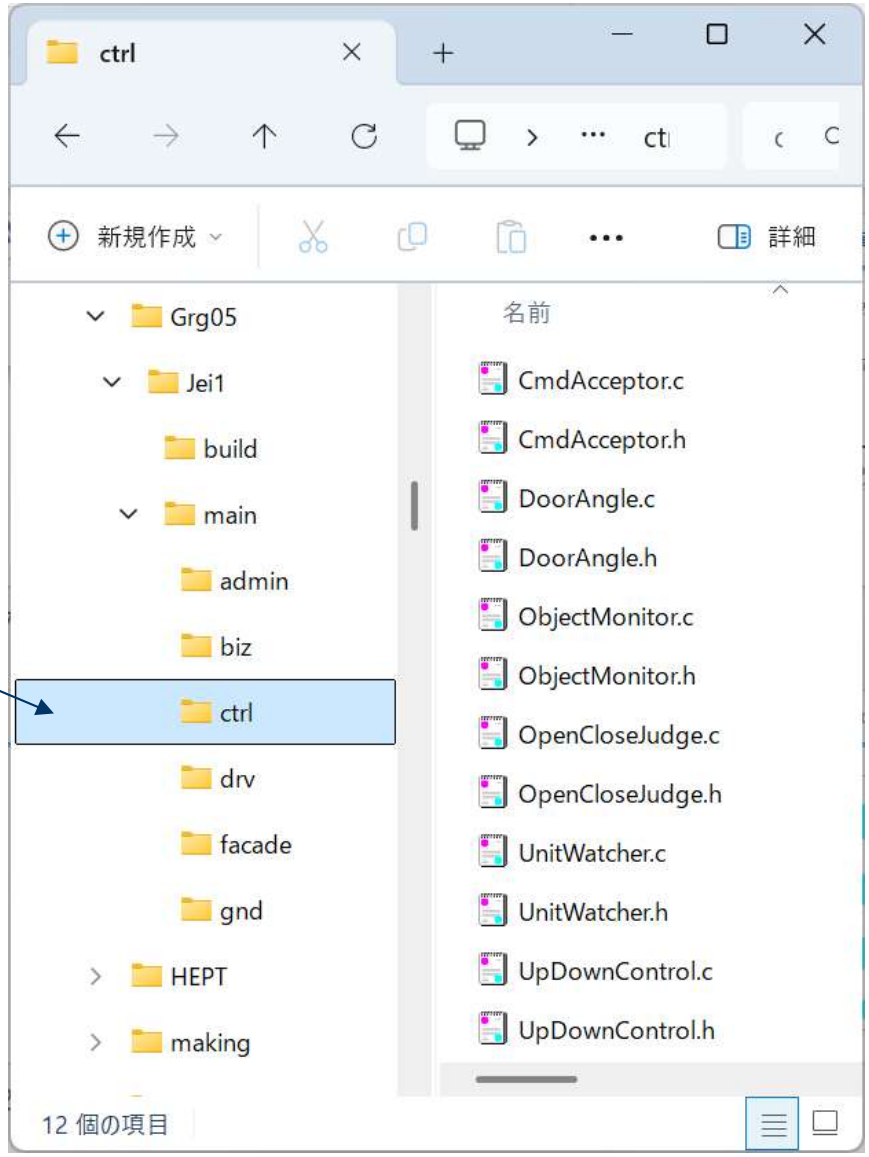
BACKSLASH DESIGN

3-1. 全体俯瞰

A2G配置のフォルダを作りファイルを移動

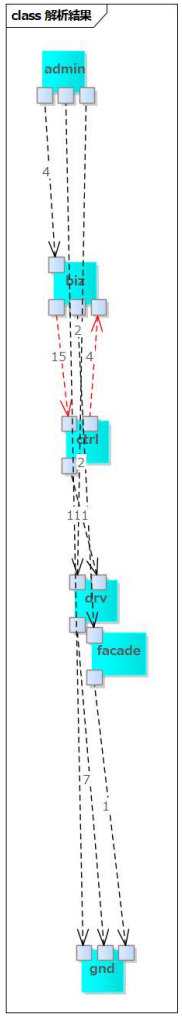
- admin-biz-ctrl-drv-façade-gndというフォルダを作り、ファイルを入れる



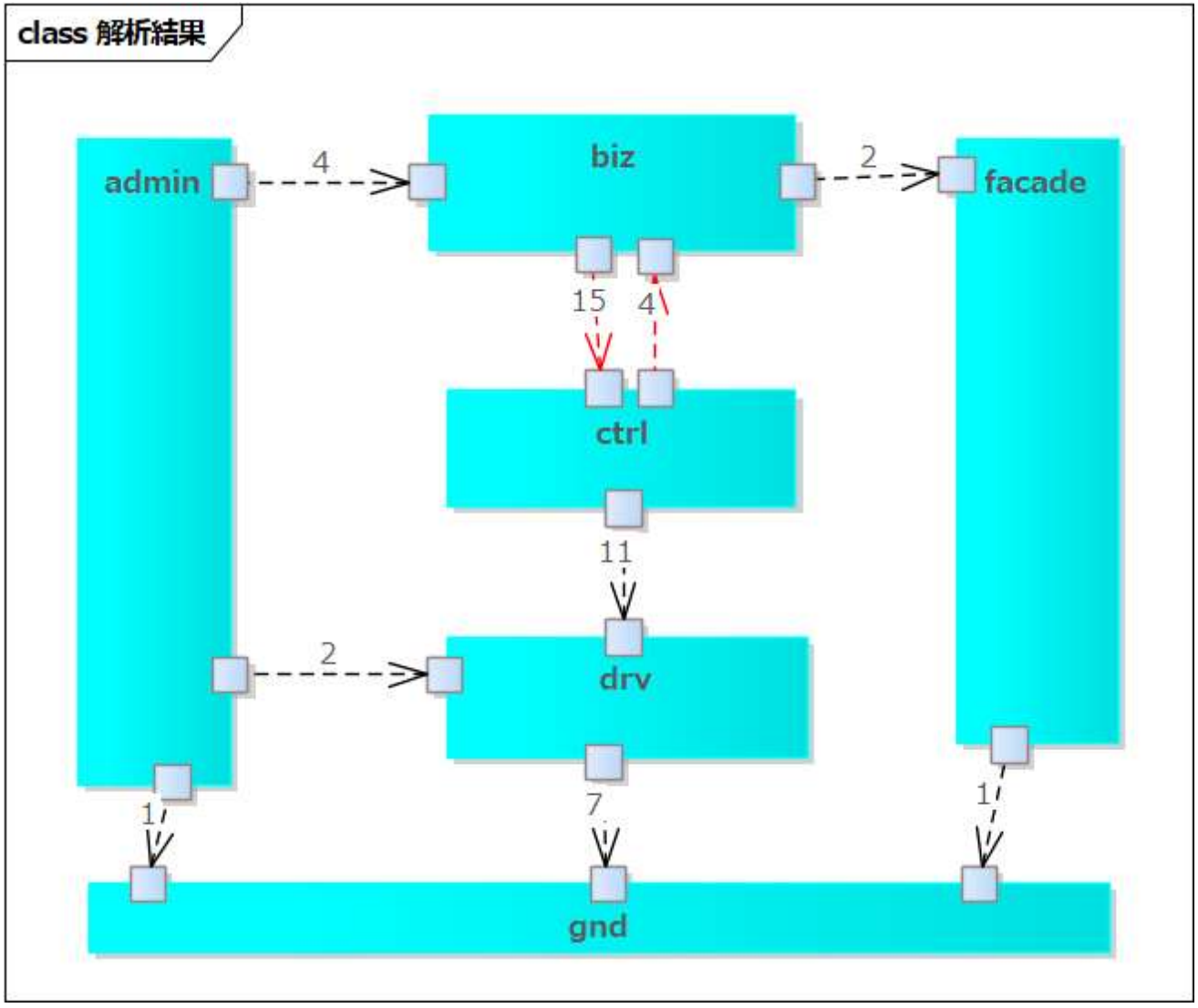


フォルダ単位のコンポーネント図を生成し、配置を変更

- AtScopeでコンポーネント図を出力



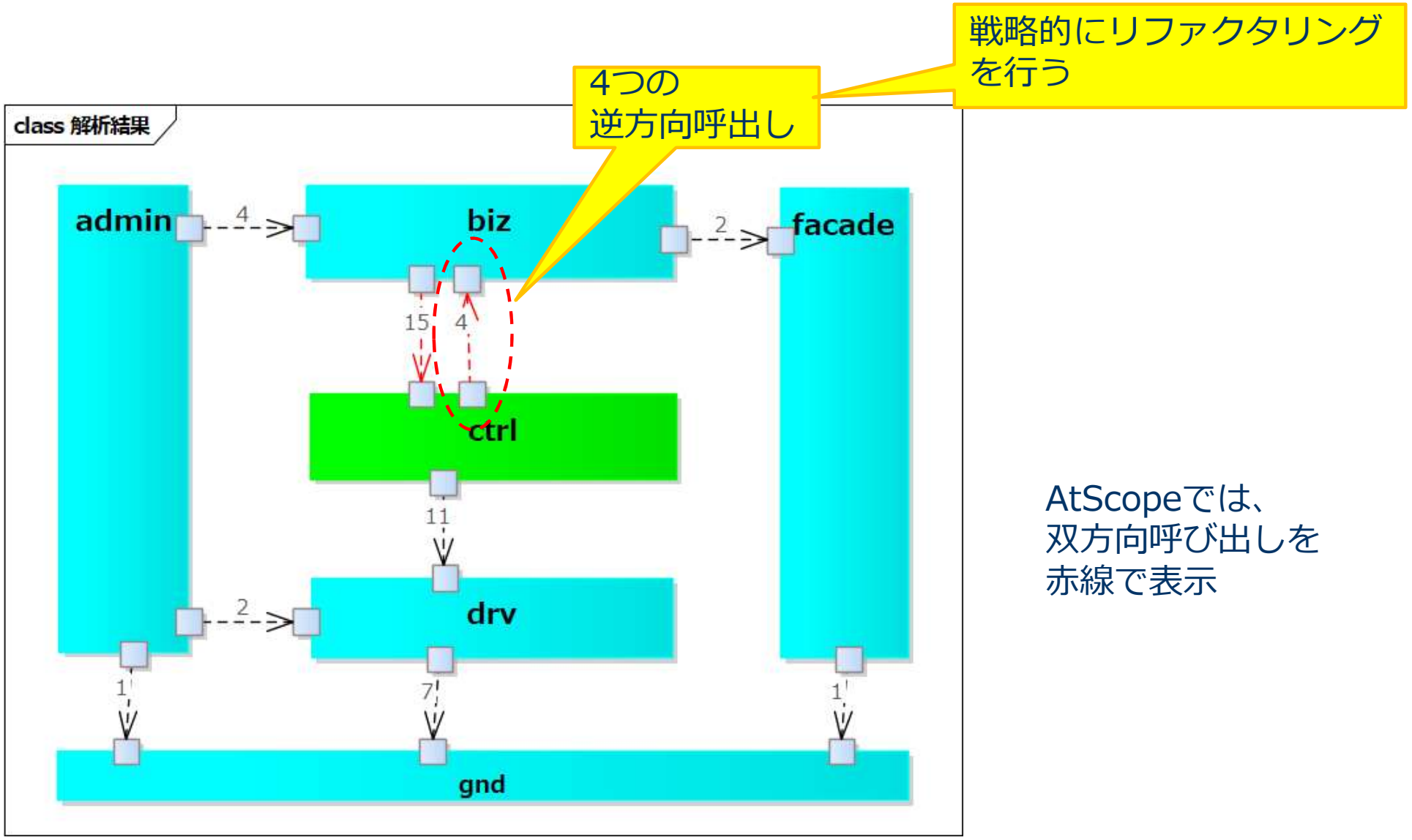
手動で配置を整える



BACKSLASH DESIGN

3-2. 単方向依存

単方向化すべき箇所を特定



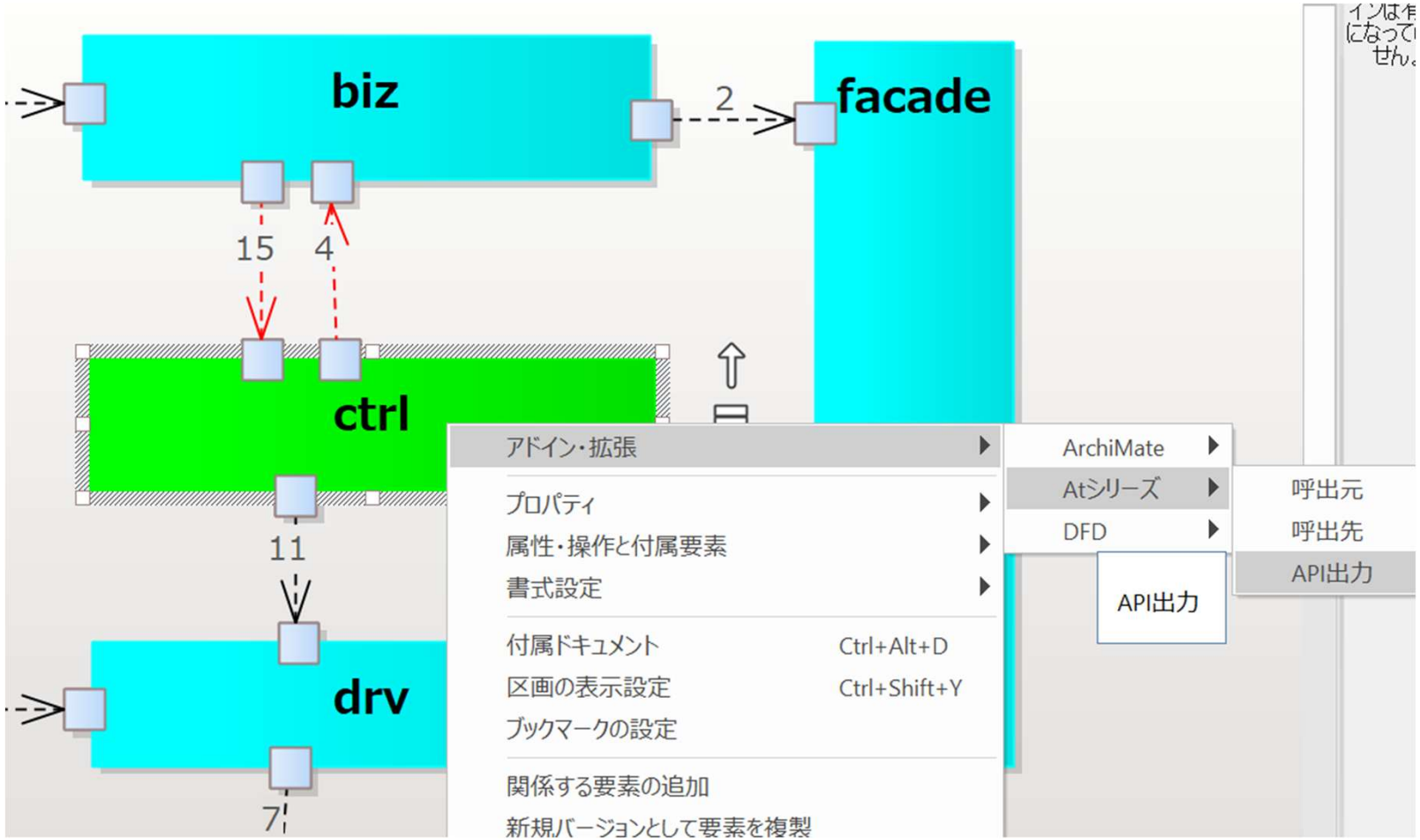
AtScopeでは、
双方向呼び出しを
赤線で表示

BACKSLASH DESIGN

3-3. インターフェイス定義

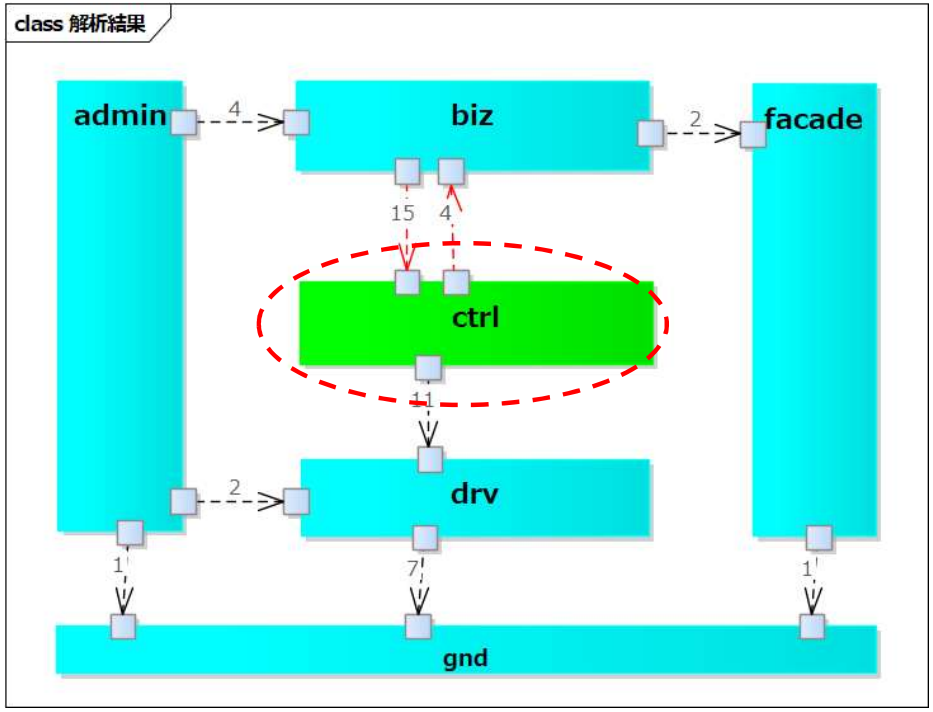
AtScopeでの関数一覧出力

- 要素を右クリック⇒「アドイン・拡張」⇒「Atシリーズ」⇒「API出力」で、呼出し一覧をcsvファイルへ出力



ctrl層のAPI一覧

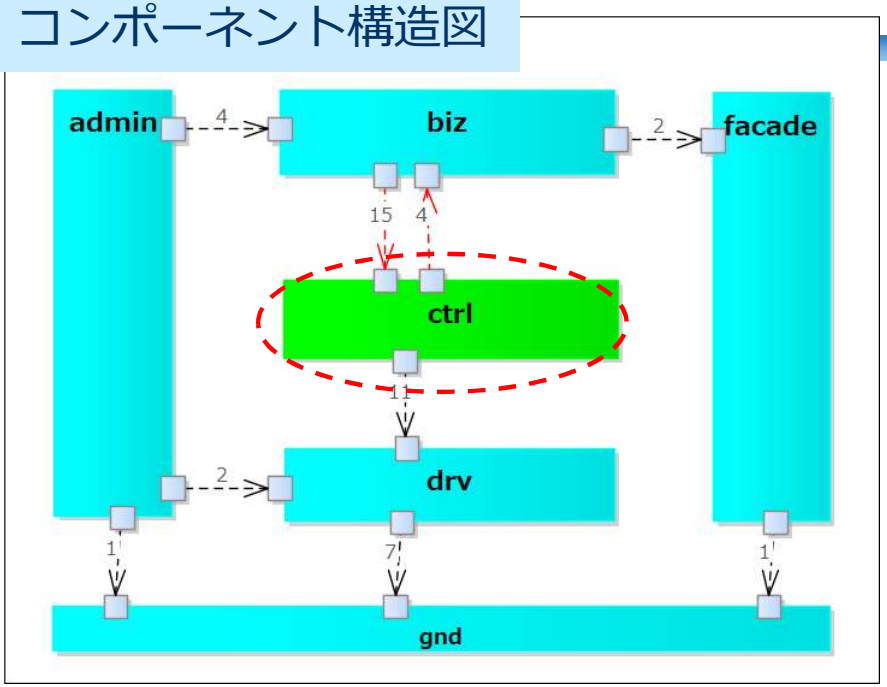
- コンポーネント間の関数呼出しを列挙
  - 整理することでコンポーネントのAPIとして定義できる



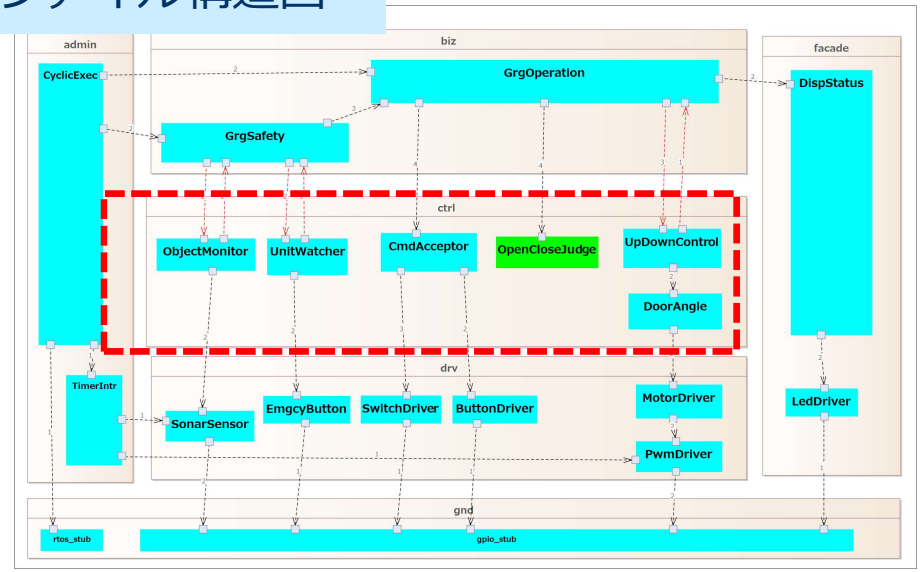
呼出元ファイル	呼出元関数	呼出先ファイル	呼出先関数
GrgOperation	go_init	CmdAcceptor	ca_init
GrgOperation	go_init	OpenCloseJudge	oj_init
GrgOperation	go_init	UpDownControl	uc_init
GrgOperation	go_run	CmdAcceptor	ca_chkLock
GrgOperation	go_run	CmdAcceptor	ca_rcvCommand
GrgOperation	go_run	OpenCloseJudge	oj_judgeOpenClose
GrgOperation	go_run	UpDownControl	uc_moveDoor
GrgOperation	go_rcvOpenClose	UpDownControl	uc_operateDoor
GrgOperation	go_ntfyReached	OpenCloseJudge	oj_reached
GrgOperation	go_stopEmergency	OpenCloseJudge	oj_stopped
GrgOperation	go_dispStatus	CmdAcceptor	ca_isLock
GrgSafety	gs_init	UnitWatcher	uw_init
GrgSafety	gs_init	ObjectMonitor	om_init
GrgSafety	gs_run	UnitWatcher	uw_monEmergency
GrgSafety	gs_run	ObjectMonitor	om_monObject
CmdAcceptor	ca_init	SwitchDriver	sd_init
CmdAcceptor	ca_init	ButtonDriver	bd_init
CmdAcceptor	ca_init	SwitchDriver	sd_readSwitch
CmdAcceptor	ca_chkLock	SwitchDriver	sd_readSwitch
CmdAcceptor	ca_rcvCommand	ButtonDriver	bd_readButton
DoorAngle	da_init	MotorDriver	md_init
DoorAngle	da_stepAngle	MotorDriver	md_rotate
ObjectMonitor	om_init	SonarSensor	smgr_init_sonar
ObjectMonitor	om_monObject	SonarSensor	smgr_get_pulsewidth
UnitWatcher	uw_init	EmgcyButton	eb_init
UnitWatcher	uw_monEmergency	EmgcyButton	eb_readEmgcyButton
ObjectMonitor	om_monObject	GrgSafety	gs_ntfyDetection
ObjectMonitor	om_monObject	GrgSafety	gs_ntfyRemoved
UnitWatcher	uw_monEmergency	GrgSafety	gs_stopEmergency
UpDownControl	uc_moveDoor	GrgOperation	go_ntfyReached

ctrl層の呼び出し関係

コンポーネント構造図



ファイル構造図



呼出元ファイル	呼出元関数	呼出先ファイル	呼出先関数
GrgOperation	go_init	CmdAcceptor	ca_init
GrgOperation	go_init	OpenCloseJudge	oj_init
GrgOperation	go_init	UpDownControl	uc_init
GrgOperation	go_run	CmdAcceptor	ca_chkLock
GrgOperation	go_run	CmdAcceptor	ca_rcvCommand
GrgOperation	go_run	OpenC	
GrgOperation	go_run	UpDo	
GrgOperation	go_rcvOpenClose	UpDow	
GrgOperation	go_ntfyReached	OpenC	
GrgOperation	go_stopEmergency	OpenCloseJudge	oj_stopped
GrgOperation	go_dispStatus	CmdAcceptor	ca_isLock
GrgSafety	gs_init	UnitWatcher	uw_init
GrgSafety	gs_init	ObjectMonitor	om_init
GrgSafety	gs_run	UnitWatcher	uw_monEmergency
GrgSafety	gs_run	ObjectMonitor	om_monObject
CmdAcceptor	ca_init	SwitchDriver	sd_init
CmdAcceptor	ca_init	ButtonDriver	bd_init
		SwitchDriver	sd_readSwitch
		SwitchDriver	sd_readSwitch
		ButtonDriver	bd_readButton
DoorAngle	da_init	MotorDriver	md_init
DoorAngle	da_stepAngle	MotorDriver	md_rotate
ObjectMonitor	om_init	SonarSensor	smgr_init_sonar
ObjectMonitor	om_monObject	SonarSensor	smgr_get_pulsewidth
UnitWatcher	uw_init	EmgcyButton	eb_init
UnitWatcher	uw_monEmergency	EmgcyButton	eb_readEmgcyButton
C		GrgSafety	gs_ntfyDetection
C		GrgSafety	gs_ntfyRemoved
U	gency	GrgSafety	gs_stopEmergency
U		GrgOperation	go_ntfyReached

呼び出されている関数が
提供インタフェース
API定義

利用している関数が
要求インタフェース

逆方向呼出し
⇒コールバック
などで単方向化

BACKSLASH DESIGN

４．まとめ：コード起点での資産化

7STEPコード起点プロダクトライン開発

BACKSLASH DESIGN

STEP	アクティビティ	要点
1	全体俯瞰	静的構造で全体を俯瞰する
2	単方向依存	双方向依存をリファクタリングして置換しやすくする
3	提供インタフェース定義	関数コールを集約してAPI定義する
4	スコーピング	プロダクトライン化する単位を決める
5	フィーチャ定義	機種ごとの変動要因を定義する
6	変動点マップ	変動要因と設計要素の組合せ表を作る
7	可変性実装	機能確定時毎に分類し設計実装する